# A Combinatorial Algorithm to Compute Regularization Paths

Bernd Gärtner
ETH Zurich, Switzerland
gaertner@inf.ethz.ch

Joachim Giesen
Friedrich-Schiller-Universität
Jena, Germany
giesen@informatik.uni-jena.de

Martin Jaggi
ETH Zurich, Switzerland
jaggi@inf.ethz.ch

Torsten Welsch
Friedrich-Schiller-Universität
Jena, Germany



## ABSTRACT

For a wide variety of regularization methods, algorithms computing the entire solution path have been developed recently. Solution path algorithms do not only compute the solution for one particular value of the regularization parameter but the entire path of solutions, making the selection of an optimal parameter much easier. Most of the currently used algorithms are not robust in the sense that they cannot deal with general or degenerate input. Here we present a new robust, generic method for parametric quadratic programming. Our algorithm directly applies to nearly all machine learning applications, where so far every application required its own different algorithm.

We illustrate the usefulness of our method by applying it to a very low rank problem which could not be solved by existing path tracking methods, namely to compute part-worth values in choice based conjoint analysis, a popular technique from market research to estimate consumers preferences on a class of parameterized options.

## Keywords

Regularization Paths, Solution Paths, Parameterized Optimization, Support Vector Machines, Kernel Methods, Conjoint Analysis

## 1. INTRODUCTION

We study a combinatorial algorithm to solve parameterized quadratic programs, i.e., to compute the whole solution path. Unlike other methods employed in machine learning, our algorithm can deal with singular objective function matrices, without perturbing the input. Regularization methods resulting in parametrized quadratic programs have successfully been applied in many optimization, classification and regression tasks in a variety of areas as for example signal processing, statistics, biology, surface reconstruction and information retrieval. We will briefly review some applications here, and we will also study another application, namely choice based conjoint analysis in more detail. Conjoint analysis comprises a popular family of techniques mostly used in market research to assess consumers' preferences on a set of options that are specified by multiple parameters, see [19] for an overview and recent developments. We will show that a regularization approach to the analysis of preference data leads to a parameterized quadratic program with a sparse, low rank positive semi-definite matrix describing the quadratic term of the objective function.

### 1.1 Contributions and Related Work

*Solution Path Algorithms in Machine Learning.* An algorithm to compute the entire regularization path of the $C$-SVM has originally been reported by Hastie et al. [20]. [10] gave such an algorithm for the LASSO, and later [26] and [25] proposed solution path algorithms for $\nu$-SVM and one-class SVM respectively. Also Receiver Operating Characteristic (ROC) curves of SVM were solved by such methods [3]. Support vector regression (SVR) is interesting as its underlying quadratic program depends on two parameters, a regularization parameter (for which the solution path was tracked by [18, 36, 26]) and a tube-width parameter (for which [35] recently gave a solution path algorithm). See also [30] for a recent overview.

As Hastie et al. [20] point out, one drawback of their algorithm for the two-class SVM is that it does not work for singular kernel matrices, but requires that in the process of the algorithm, all occurring principal minors of the kernel matrix need to be invertible. The same is required by the other existing path algorithms mentioned above. However, large kernel matrices do often have very low numerical rank, even in those cases where radial base function kernels are used [20, Section 5.1], but of course also in the case of linear SVMs with sparse features, such as in the application to conjoint analysis discussed in this paper. The inability to deal with singular sub-matrices is probably one of the main reasons that none of the above mentioned algorithms could so far effectively be applied on medium/larger scale problems [20, 30]. [30, Section 4.2] report that their algorithm prematurely terminates on $3 \times 3$ matrices due to this described problem.

By observing that all the above mentioned algorithms are reporting the solution paths of parametric quadratic programming of the form (1), we point out that it is in fact not necessary to use different algorithms for each problem variant. Generic algorithms have been known for quite some

time [27, 29], [4, 5, 17], [33], but have interestingly not yet received broader attention in the area of machine learning.

One goal of this paper is to popularize the generic solution algorithms for parametric quadratic programming, because we think that they have some major advantages:

- The same algorithm can be applied to any solution path problem that can be written in the form (1), which includes all of [20, 10, 18, 36, 3, 26, 25, 11, 35].
- Many of the known generic algorithms can deal with *all* inputs; in particular the algorithms can cope with singular sub-matrices in the objective function.
- There is significant existing literature on the performance, numerical stability, and complexity of the generic algorithms.
- Our criss-cross algorithm is numerically more stable, and also more robust in the sense that small errors do not add up while tracking the solution path. Also, such algorithms are faster for sparse problems as in linear SVMs and conjoint analysis, because they do not need any matrix inversions.

*Comparison with other ways to deal with degeneracies.* Instead of using our described generic criss-cross method, another obvious way to avoid degeneracies caused by singular sub-matrices in the objective function is to add a small value $\varepsilon$ to each diagonal entry of the original matrix $Q$; subsequently, all simple methods for the regular case such as [20, 30] can be used. There are several problems with this approach. First of all, the rank of the objective function matrix is blown up artificially, and the potential of using efficient small-rank-QP methods would be wasted. Secondly, the solution path of the perturbed problem may differ substantially from that of the original problem; in particular, the perturbation may lead to a much higher number of bends and therefore higher tracking cost, and the computed solutions could be far off the real solutions. In contrast, our criss cross method avoids all these issues, since it always solves the original unperturbed problem.

# 2. PARAMETRIC QUADRATIC PROGRAMMING

A quadratic program (QP) is the problem of minimizing a convex quadratic function subject to linear equality and inequality constraints. Here, we are interested in *parameterized* quadratic programs (pQPs) of the (standard) form

$$\mathbf{QP}(\mu) \quad \begin{array}{ll} \text{minimize}_x & x^T Q x + c(\mu)^T x \\ \text{subject to} & Ax \geq b(\mu) \\ & x \geq 0, \end{array} \tag{1}$$

where $c : \mathbb{R} \to \mathbb{R}^n$ and $b : \mathbb{R} \to \mathbb{R}^m$ are functions that describe how the linear term of the objective function and the right-hand side of the constraints vary with some real parameter $\mu$. $Q$ is an $n \times n$ symmetric positive semidefinite (PSD) matrix (the quadratic quadratic term of the objective function), $c$ is an $n$-vector (the linear term of the objective function), $A$ is an $m \times n$ matrix (the constraint matrix), and $b$ is an $m$-vector (the right-hand side of the constraints).

Our goal is to solve a given problem QP($\mu$) of the form (1) for all $\mu$ in a given interval $[\mu_{\min}, \mu_{\max}]$, where we assume for now that QP($\mu$) has an optimal solution for all $\mu$ in that interval (the general case is easy to handle as well, see the remark in the "Odds and Ends" paragraph below). In other words, given any value of $\mu \in [\mu_{\min}, \mu_{\max}]$, we want to retrieve an optimal solution $x^*$ to QP($\mu$) quickly, without having to solve the problem from scratch. The task of solving such a problem for all possible values of the parameter $\mu$ is called *parametric quadratic programming*. What we want as output is a *solution path*, an explicit function $x^* : \mathbb{R} \to \mathbb{R}^n$ that describes the solution as a function of the parameter $\mu$.

It is well known that the solution path $x^*$ is piecewise linear if $c$ and $b$ are linear functions of $\mu$, see for example [28].

## 2.1 Regularization Methods and pQPs

A variety of machine learning methods, in particular many regularization methods, are direct instances of parametric quadratic programming. Examples include support vector machines [6], support vector regression [32], the LASSO [34], surface reconstruction [31], $\ell_1$-regularized least squares [23], and compressed sensing [13].

Let us shortly describe the support vector machine as a popular example of a pQP that results from regularization. Later we will re-discover the corresponding pQP in the context of choice based conjoint analysis.

*Support Vector Machine.* The support vector machine (SVM) is a standard tool for two-class classification problems. In Section 4 we will see that estimating part-worth values in choice based conjoint analysis can be seen as a problem that is geometrically dual to binary classification. The primal soft margin $C$-SVM is the following pQP:

$$\begin{array}{ll} \text{minimize}_{w,b,\xi} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^n \xi_i \\ \text{subject to} & y_i(\omega^T x_i + b) \geq 1 - \xi_i, \end{array} \tag{2}$$

where $y_i \in \{\pm 1\}$ is the class label of data point $x_i$ and $C$ is the regularization parameter. The dual of the soft margin $C$-SVM is the following pQP (observe that the regularization parameter moves from the objective function to the constraints):

$$\begin{array}{ll} \text{maximize}_\alpha & \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ \text{subject to} & \sum_i y_i \alpha_i = 1 \\ & 0 \leq \alpha_i \leq C \end{array} \tag{3}$$

# 3. THE CRISS-CROSS METHOD FOR PQPS

Next we will present a new generic algorithm that uses LCP techniques; in contrast to Murty's method [27], it uses the extremely simple and elegant *criss-cross method* as a subroutine, resulting in what we believe is the simplest generic algorithm that is able to deal with arbitrary PSD matrices $Q$.

The algorithm works in principle for more general continuous functions $c$. The main idea is to transform (1) to a parametric *linear complementarity problem* (LCP), and then use the *criss-cross method* to quickly update the solution while $\mu$ varies.

## 3.1 The LCP Formulation

Let us recall the Karush-Kuhn-Tucker optimality conditions for quadratic programs, see e.g. [8, Section 2.8].

THEOREM 1. *An $n$-vector $x$ is an optimal solution to (1) if and only if there exists and $n$-vector $u$ as well as $m$-vectors $y$ and $v$ such that*

$$\begin{array}{lrcl} (i) & v = Ax - b(\mu) \geq 0 & \text{and} & x \geq 0 \\ (ii) & u = c(\mu) - A^T y + 2Qx \geq 0 & \text{and} & y \geq 0 \\ (iii) & x^T u = 0 & \text{and} & y^T v = 0, \end{array}$$

where (i) encodes primal feasibility of $x$, (ii) encodes dual feasibility of $y$, and (iii) is referred to as complementary slackness.

The three conditions of the previous theorem can be rewritten in the form

$$\begin{array}{rcl} w - Mz & = & q(\mu) \\ w, z & \geq & 0 \\ w^T z & = & 0, \end{array} \tag{4}$$

where $w^T = (u^T, v^T)$, $z^T = (x^T, y^T)$, $q(\mu)^T = (c(\mu)^T, -b(\mu)^T)$ and $M = \begin{pmatrix} 2Q & -A^T \\ A & 0 \end{pmatrix}$.

Problem (4) is a *linear complementarity problem* (LCP) with a PSD matrix (for all $w$, we have $w^T M w = 2u^T Q u \geq 0$ — this is why we have chosen the constraints to be "$Ax \geq b$" instead of the more common "$Ax \leq b$"; the latter would lead to a symmetric but not necessarily positive semidefinite matrix $M$ in the LCP (4)). In order so solve (1), we will therefore find $w^*$ and $z^*$ that satisfy (4); then the first $n$ components of the $(n+m)$-vector $z^*$ form a solution to (1). This reduction of QP to LCP is well-known, see e.g. [8, Section 1.2].

## 3.2 The Criss-Cross Method

The criss-cross method is a combinatorial method for finding vectors $w$ and $z$ that satisfy (4), given that $q = q(\mu)$ is fixed (we address the case of varying $\mu$ below). The method is guaranteed to terminate (with a solution, or a proof of infeasibility of (4)), given that $M$ is a *sufficient* matrix, see e.g. [14]. This matrix class contains all PSD matrices, meaning that the criss-cross method is applicable in our setting. Our description below is for the special case of PSD matrices [24].

The criss-cross method is an iterative method that goes through a sequence of *basic* solutions. To define such a solution, we consider any subset $B \subseteq [k], k := n + m$ and the matrix $M_B$ whose $j$-th column is the $j$-th column $I_j$ of the $k \times k$ identity matrix $I$ (if $j \in B$), or the $j$-th column of $-M$ (if $j \notin B$). $B$ is called a *basis* if $M_B$ is invertible. For example, $B = [k]$ is a basis since $M_{[k]} = I$.

Given a basis, we obtain the corresponding *basic solution* as the unique solution of the following system of equations:

$$\begin{array}{rcll} z_j & = & 0, & j \in B, \\ w_j & = & 0, & j \notin B, \\ w - Mz & = & q. & \end{array}$$

This indeed has a unique solution, since substitution of the first two sets of equations into $w - Mz = q$ yields the system $M_B \lambda_B = q$, where $\lambda_j = w_j$ if $j \in B$ and $\lambda_j = z_j$ otherwise.

It is clear that every basic solution $(w, z)$ satisfies $w^T z = 0$, but $w, z \geq 0$ may not hold. The criss-cross method tries to rectify this by repeatedly moving to another basis and corresponding basic solution, until $w, z \geq 0$ in which case the LCP is solved.

Given a basis $B$ along with $\lambda_B^*$ (the unique solution of $M_B \lambda_B = q$), one step of the method works as follows. If $\lambda^* := \lambda_B^* \geq 0$, we are done; otherwise, choose the smallest index $r$ such that $\lambda_r^* < 0$. With respect to $B$, the system $w - Mz = q$ can be written as $M_B \lambda_B + M_N \lambda_N = q$, where $N = [k] \setminus B$. Consequently,

$$\lambda_B = M_B^{-1} q - M_B^{-1} M_N \lambda_N$$

for all solutions of $w - Mz = q$ (the basic solution associated with $B$ is obtained from $\lambda_N = 0$).

Let the $k \times k$ matrix $\Lambda = -M_B^{-1} M_N$ be the *dictionary* associated with $B$, so that we have

$$\lambda_B = M_B^{-1} q + \Lambda \lambda_N. \tag{5}$$

There are now two cases:

(a) $\Lambda_{rj} \leq 0$ for all $j \in [k]$. By (5) we have

$$(\lambda_B)_r = \lambda_r^* + \Lambda^r \lambda_N$$

for all solutions of $w - Mz = q$, where $\Lambda^r$ is the $r$-th row of $\Lambda$. But since this yields $\lambda_r < 0$ whenever $\lambda_N \geq 0$, there can't be any solution to $w - Mz = q$ with $w, z \geq 0$, and we can conclude that the LCP is infeasible.

(b) $\Lambda_{rj} > 0$ for some $j \in [k]$. Choose the smallest index $s$ such that $\Lambda_{rs} > 0$ and set $p := \max(r, s)$. If $\Lambda_{pp} \neq 0$, update $B$ to $B' := B \oplus \{p\}$ (*diagonal pivot*), otherwise update $B$ to $B' := B \oplus \{r, s\}$ (*exchange pivot*), where $\oplus$ denotes symmetric set difference.

LEMMA 2. *The set $B'$ resulting from step (b) is again a basis.*

PROOF. In general, if $B' = B \oplus D$, then $M_{B'}$ is obtained from $M_B$ by replacing the columns whose indices are in $D$ with the corresponding columns of $M_N$. This update can be written as

$$M_{B'} = M_B T,$$

where $T_j = I_j$ if $j \notin D$ and $T_j = (M_B^{-1} M_N)_j = -\Lambda_j$ for $j \in D$. Moreover, since $M_B$ was invertible, $M_{B'}$ is invertible if and only if $\det(T) \neq 0$. If $D = \{p\}$ (the diagonal pivot), we get $\det(T) = -\Lambda_{pp} \neq 0$. If $D = \{r, s\}$ (the exchange pivot), we assume w.l.o.g. $r < s = p$ and get

$$\det(T) = \det \begin{pmatrix} \Lambda_{rr} & \Lambda_{rs} \\ \Lambda_{sr} & 0 \end{pmatrix}.$$

In order to evaluate this, we need one observation concerning the structure of $\Lambda = -M_B^{-1} M_N$. Let us call an $n \times$

$n$ matrix *bisymmetric* if it is of the form $\begin{pmatrix} Q & -A^T \\ A & P \end{pmatrix}$ where both $Q$ and $P$ are symmetric. For example, $M = -M_{[k]}^{-1} M_\emptyset$ is bisymmetric, but simple calculations show that $\Lambda = -M_B^{-1} M_N$ is also bisymmetric, hence $\Lambda_{sr} = -\Lambda_{rs} < 0$ which implies $\det(T) > 0$. $\square$

This method is due Klafszky and Terlaky [24] who also show that it terminates after having gone through a finite number of bases.

## 3.3 Varying the Parameter

We now turn to the case where the right-hand side $q(\mu)$ of (4) varies. Assume that we have solved the problem for $\mu = \mu_{\min}$ using the criss-cross method, meaning that we now have a basis $B \subseteq [k]$ such that

$$\lambda_B^*(\mu) = M_B^{-1} q(\mu) \geq 0.$$

Since $\lambda_B^*(\mu)$ depends linearly on $\mu$ (assuming that $b$ and $c$ in (1) are linear functions), we can easily compute the largest value $\mu' \geq \mu$ such that $\lambda_B^*(\mu') \geq 0$ (we may have $\mu' = \mu$ but also $\mu' = \infty$).

For every value $\kappa \in [\mu, \mu']$, $\lambda_B^*(\kappa)$ is still a solution to (4) with right-hand side $q(\kappa)$. In order to be able to trace the solution beyond $\kappa = \mu'$, we apply the criss-cross method to (4) again, starting from the basis $B$, but now with the right-hand side $q = q(\mu' + \varepsilon)$, where $\varepsilon$ is a symbolic parameter meant to represent an arbitrarily small positive value. That way, we solve a slightly perturbed LCP, starting from a solution to the old LCP, and in practice, we expect that this will take only very few iterations. There are no theoretical guarantees for this, though[1].

In running the criss-cross-method on the symbolically perturbed problem, all values $\lambda_r^*$ whose signs are being used to check whether we currently have a solution to (4) are linear polynomials in $\varepsilon$ (dictionary entries that are needed to check for infeasibility are unaffected by $\varepsilon$). The sign of a linear polynomial $\sigma + \varepsilon\tau$ is determined by $\sigma$ if $\sigma \neq 0$, and by $\tau$ otherwise.

It follows that for the basis $B'$ obtained upon termination of the criss-cross method, there are $k$-vectors $s$ and $t$ such that

$$\lambda_{B'}^*(\mu' + \varepsilon) = s + \varepsilon t, \tag{6}$$

where $s_j > 0$ or $s_j = 0, t_j \geq 0 \quad \forall j \in [k]$.

This implies that $\lambda_{B'}^*(\mu' + \varepsilon) \geq 0$ for any sufficiently small *numerical* value of $\varepsilon$. In other words, $B'$ is valid throughout a whole interval $[\mu', \mu' + \varepsilon']$, where $\varepsilon' > 0$ is easy to compute from (6).

While increasing $\mu$, we therefore subdivide our interval $[\mu_{\min}, \mu_{\max}]$ into pieces over which the solution to (4) and therefore also the solution to (1) is linear in $\mu$. There are only finitely many such pieces, since no basis $B$ can repeat (if $B$ is valid for two values $\mu, \mu'$, it is also valid for any intermediate value).

[1]This complexity behavior is expected to be very similar to running Simplex steps for a slightly perturbed linear program, starting from a solution for the original problem.

*Performance.* By the above analysis, we have that our algorithm calculates the entire solution path of any parametric quadratic program in finite time. Also, it is well suited to make use of the sparseness of the solutions, which is a key property of all regularization methods. When running the algorithm, the relevant size of the matrices $M_B$ that we have to deal with is bounded by the number the number of non-zero entries in $x$, plus $m$.

*Odds and Ends.* The solution path computed in the above way may be discontinuous, since the solution to the LCP may "jump" when we move from $q(\mu')$ to $q(\mu' + \varepsilon)$. This is due to the fact that the LCP has in general not a unique solution, and the criss-cross method has no control over which optimal solution it finds. However if one strictly wants continuity, one can simply insert connecting straight-line segments: Since both endpoints are solutions for $q(\mu')$ (set $\varepsilon = 0$), all intermediate points will be solutions as well. This holds for the $x$-part of $(w, z)$ (the QP solution) by convexity of the optimal region in (1), but it also holds for $(w, z)$ w.r.t. the LCP by a result of Adler and Gale [2].

For the above to work, we do not even have to assume that $\mathrm{QP}(\mu)$ has an optimal solution throughout $[\mu_{\min}, \mu_{\max}]$. Our method can handle the general case. We may start off at $\mu = \mu_{\min}$ with an unsolvable LCP (the criss-cross method will report this), or we may run into an unsolvable situation later. In order to trace $\mu$ through such a situation, we simply choose the "next event" as the largest $\mu' \geq \mu$ for which $(\lambda_B^*)(\mu')_r \leq 0$, where $(\lambda_B)_r$ is the variable for which infeasibility was detected in case (a) of the criss-cross method.

# 4. CHOICE BASED CONJOINT ANALYSIS

In general conjoint analysis includes two tasks: (a) preference data assessment, and (b) analysis of the assessed data. In choice based conjoint analysis (CBC) preference data are assessed on a set of options $A$ in a sequence of choice experiments. In every choice experiment a consumer has to choose the most preferred option out of a few options that are presented to her/him (typically between two and five options from $A$). The set of all options is assumed to carry a conjoint structure, i.e., $A$ is the Cartesian product $A = A_1 \times \ldots \times A_n$ of parameter sets $A_i$. In the following we assume that the parameter sets $A_i$ are finite. Choice data are of the form $a \succeq b$, where $a = (a_1, \ldots, a_n), b = (b_1, \ldots, b_n) \in A$ and $a$ was preferred over $b$ by some consumer in a choice experiment. Our goal is to compute an interval scale $v : A \to \mathbb{R}$ on the domain $A$ from a set of choice data. The scale $v$ is meant to represent the preferences of the population of consumers who contributed to the choice experiments, i.e., $a \in A$ is more popular than $b \in A$ if $v(a) > v(b)$, and the difference $v(a) - v(b)$ tells how much more popular $v(a)$ is than $v(b)$.

In the data analysis stage of conjoint analysis it is almost always assumed [1, 7] that the scale $v$ is linear, i.e., that it can be written as as

$$v(a) = v\big((a_1, \ldots, a_n)\big) = \sum_{i=1}^{n} v_i(a_i), \tag{7}$$

where $v_i : A_i \to \mathbb{R}$ are also interval scales, see [22] for a

justification of choosing a linear scale. The value $v_i(a_i), a_i \in A_i$ is called the part-worth value of level $a_i$, i.e., the value that it contributes to the overall value of an option $a$ where the level $a_i$ is present. The goal of choice based conjoint analysis is to compute/estimate part-worth values for all attribute levels from choice data.

*Regularization approach to compute part-worth values.* Our goal here is to review how computing part-worth values in choice based conjoint analysis naturally leads to a *geometrically dual* formulation of a SVM, see [15] for more details. Assuming that the scale $v$ is linear, then the part-worth values $v_i(a_{ij}) \in \mathbb{R}, a_{ij} \in A_i, i = 1, \ldots, n$, should satisfy constraints of the form

$$\sum_{i=1}^{n} v_i(a_i) - v_i(b_i) > 0, \tag{8}$$

whenever $a = (a_1, \ldots, a_n)$ was preferred over $b = (b_1, \ldots, b_n)$ by some consumer in a choice experiment. Let $m_i = |A_i|$ and $m = \sum_{i=1}^{n} m_i$. Any linear scale $v$ on the domain $A$ is represented by a vector $\big(v_i(q_{ij})\big)_{i=1,\ldots,n;j=1,\ldots,m_i} \in \mathbb{R}^m$, and a choice experiment is defined by the characteristic vectors $\chi_a \in \{0,1\}^m$, whose $i$'th component is 1 if the corresponding parameter level is present in option $a$, and 0 otherwise. We can re-write the choice constraints (8) as

$$v^t(\chi_a - \chi_b) > 0, \text{ if } a \succeq b \text{ in a choice experiment,}$$

or shorter as $v^t n_{ab} > 0$, where $n_{ab} = (\chi_a - \chi_b)$. A vector $v$ is called *feasible* if it satisfies all constraints. The set of all feasible vectors is in general a (not necessarily full-dimensional) double cone whose apex is the origin. Among all the feasible vectors we want to choose one with good generalization properties. This can be phrased as a two-class classification problem as follows: let $H_{ab}$ be the hyperplane

$$H_{ab} = \{v \in \mathbb{R}^m \mid v^t n_{ab} = 0\}$$

with normal $n_{ab}$, and let

$$H_{ab}^{+} = \{v \in \mathbb{R}^m \mid v^t n_{ab} \geq 0\}$$

and

$$H_{ab}^{-} = \{v \in \mathbb{R}^m \mid v^t n_{ab} \leq 0\}$$

be the two closed halfspaces bounded by $H_{ab}$. Note that $H_{ab}^{+} = H_{ba}^{-}$. If $a$ was preferred over $b$ in a choice experiment, then we have a constraint of the form $v \in H_{ab}^{+}$, otherwise, if $b$ was preferred over $a$ in a choice experiment, then we have $v \in H_{ab}^{-}$. That is, we can assign a label $+$, or $-$, respectively to the hyperplane $H_{ab}$ depending on the outcome of a choice experiment for this hyperplane. Since the label attached to the hyperplane $H_{ba}$ is just the opposite of the label attached to $H_{ab}$ we can restrict ourselves to one of the two hyperplanes for every pair $a \neq b \in A$, e.g., by fixing an arbitrary order on the elements of $A$, and only considering hyperplanes $H_{ab}$, where $a$ comes before $b$ in this order. That is, we are given labelled hyperplanes as input and are looking for a point in the feasible cone that can be written as the intersection of the halfspaces $H_{ab}^{+}$ if $a \succeq b$ in a choice experiment, and $H_{ab}^{-}$ if $b \succeq a$ in a choice experiment. In standard linear two-class classification the situation is the other way around: we are given labelled points and are looking for a hyperplane that separates the points according to their labels. There are several geometric duality transform know that map hyperplanes into points and vice versa, see for example [9], which in principle allow to transform our problem to compute part-worth values into a standard two-class classification problem. The duality transform that we consider here maps non-vertical (labeled) hyperplanes to (labeled) points and vice versa see Figure 1 for an example in $\mathbb{R}^2$. Since many of the hyperplanes $H_{ab}$ are vertical, i.e., parallel to the $m$'th coordinate axis, we augment the hyperplane normals with a $(m+1)$'th coordinate and set the value of this coordinate to $\epsilon > 0$. This leads to a two-class classification problem that is parameterized by $\epsilon$. Formulating the SVM for this problem and taking the limit $\epsilon \to 0$ leads to the following QP:

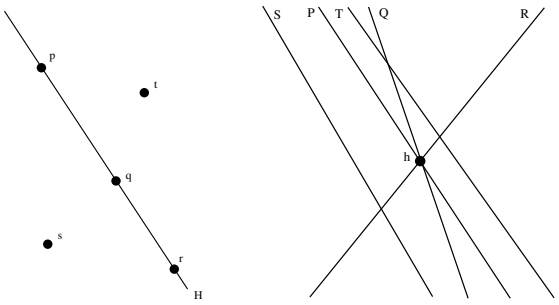


**Figure 1: An important property of duality that makes it useful for our application is that the duality of non-vertical hyperplanes and points preserves relative positions. Dual points are labeled by lowercase letters, and dual hyperplanes by capital letters.**

$$\begin{array}{lll} \textbf{CBC} & \text{minimize}_v & \frac{1}{2}\|v\|^2 \\ & \text{subject to} & v^t n_{ab} \geq 1, \text{ if } a \succeq b \\ & & \quad \text{in a choice experiment.} \end{array}$$

*Soft margin formulation.* On real data we have to deal with contradictory information, i.e., observed choices of the form $a \succeq b$ and $b \succeq a$, especially when assessing preferences on a population, but also individuals can be inconsistent in their choices. Also with contradictory information we can proceed as before with the only difference that after dualizing we work with a soft margin $C$-SVM (2) to deal with the contradictions. This leads to the following pQP:

$$\begin{array}{lll} \textbf{CBC}(\mathrm{C}) & \text{minimize}_{v,z} & \frac{1}{2}\|v\|^2 + C\sum_{j=1}^{m}\xi_j \\ & \text{subject to} & v^t n_{ab} \geq 1 - \xi_j, \text{if} \\ & & \quad a \succeq b \text{ in the } j\text{'th choice} \\ & & \quad \text{experiment.} \\ & & \xi_j \geq 0, \end{array}$$

with a non-negative slack variable $\xi_j$ for every choice, i.e., a constraint $v(a) - v(b) + \xi_j \geq 0, \xi_j \geq 0$ if $a$ was preferred over $b$ in the $j$'th choice experiment. The slack is penalized in the objective function by the term $\sum_{j=1}^{k}\xi_j$ assuming that we have information from $k$ choice experiments, and $C > 0$ is the standard trade-off parameter between model complexity and quality of fit on the observed data. The problem $\mathbf{CBC}(C)$ has already been suggested by Evgeniou

et al. [12] to compute part-worth values, but without giving details why it is well suited for that task. A similar resulting formulation is also known as the *ranking SVM* [21], when the representing features are the present parameter levels in each option.

## 5. EXPERIMENTAL RESULTS

To test the criss-cross method we provided an experimental proof of concept implementation. The implementation at its current stage is not really efficient, but the results for the number of iterations needed by the criss-cross method along the path are promising. We hope to fully exploit this behavior with a state of the art implementation in the near future.

We tested the criss-cross method on a choice based conjoint analysis data set that we obtained in a larger user study to measure the perceived quality for a visualization task [16]: the conjoint study had six parameters with $3, 5, 6, 2, 3$, and 5 respectively levels. That is, in total this study comprised 24 levels for which we estimate the part-worth value. To estimate the part-worth values we had over participants of our study had to provide answers in choice experiments. Hence the problem $\mathbf{CBC}(C)$ leads to problem $\mathbf{QP}(\mu)$ whose matrix $Q$ has dimension $(24+s)\times(24+s)$, and whose matrix $A$ has dimension $s\times(24+s)$, when $s$ is the number of choice experiments. Hence we can essentially control the complexity of the problem by the number of choice experiments considered.

For 40 choice experiments exemplary paths need $(580, 47, 22)$ or $(580, 4, 48)$ iterations for the criss-cross method for the first three bends on a $C$-interval from $[1, 6.0910^{13}]$. This clearly shows that even if a starting point for the solution path may needs some time to be computed (580 steps by the criss-cross method, but of course other methods could also be used to compute a starting solution) our described criss-cross method is very effective to continue the path at the bends.

## 6. CONCLUSION

We have presented a generic solution path algorithm for parameterized quadratic programs that works for all regularization methods that result in a single parametric quadratic program, also when the kernel matrix is not of full rank.

Since the state of the art solution methods in machine learning are moving away from finding exact solutions to faster approximate methods, it would be an interesting further research topic to investigate paths of approximate solutions of parametrized quadratic programs. Also, it should be further investigated how multi-parametric programming approaches [33] may help to find several parameters simultaneously, such as the regularization parameter, regression tube width, and also kernel parameters [37].

## 7. REFERENCES


[1] http://www.sawtoothsoftware.com.

[2] I. Adler and D. Gale. On the solutions of the positive semi-definite complementarity problem. *Technical Report*, 1975.

[3] F. Bach, D. Heckerman, and E. Horvitz. Considering cost asymmetry in learning classifiers. *The Journal of Machine Learning Research*, 7, 2006.

[4] R. Benveniste. One way to solve the parametric quadratic programming problem. *Mathematical Programming*, 21(1):224–228, 1981.

[5] M. J. Best. An algorithm for the solution of the parametric quadratic programming problem. *Applied mathematics and parallel computing - Festschrift for Klaus Ritter, H.Fischer, B.Riedemller and S.Schffler (editors), Heidelburg: Physica-Verlag*, pages 57–76, 1996.

[6] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[7] O. Chapelle and Z. Harchaoui. A machine learning approach to conjoint analysis. In *NIPS*, 2004.

[8] R. Cottle, J. S. Pang, and R. E. Stone. *The Linear Complementarity problem*. 1992. Academic Press.

[9] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer, 1987.

[10] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 2004.

[11] M. Eigensatz, J. Giesen, and M. Manjunath. The solution path of the slab support vector machine. *CCCG '08: Proceedings of the 20th Canadian Conference on Computational Geometry*, pages 211–214, 2008.

[12] T. Evgeniou, C. Boussios, and G. Zacharia. Generalized robust conjoint estimation. *Marketing Science*, 24(3):415–429, 2005.

[13] M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586 – 597, 2007.

[14] K. Fukuda, M. Namiki, and A. Tamura. Ep theorems and linear complementarity problems. *Discrete Applied Mathematics*, 84(1-3):107–119, 1998.

[15] J. Giesen. Mansucript. 2008.

[16] J. Giesen, K. Mueller, E. Schuberth, L. Wang, and P. Zolliker. Conjoint analysis to measure the perceived quality in volume rendering. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1664–1671, 2007.

[17] D. Goldfarb and K. Scheinberg. On parametric semidefinite programming. *Applied Numerical Mathematics*, 29(3):361–377, 1999.

[18] L. Gunter and J. Zhu. Computing the solution path for the regularized support vector regression. *NIPS '05: Advances in Neural Information Processing Systems 18*, 2005.

[19] A. Gustafsson, A. Herrmann, and F. Huber. Conjoint analysis as an instrument of market research practice. In A. Gustafsson, A. Herrmann, and F. Huber, editors, *Conjoint Measurement. Methods and*

*Applications*, pages 5–45. Springer, Berlin, 2000.

[20] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *The Journal of Machine Learning Research*, 5:1391 – 1415, 2004.

[21] T. Joachims. Optimizing search engines using clickthrough data. *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, Jan 2002.

[22] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-Offs*. Cambridge University Press, Cambridge, 1993.

[23] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):606 – 617, 2007.

[24] E. Klafszky and T. Terlaky. Some generalizations of the criss-cross method for quadratic programming. *Tech Report / Optimization*, 24(1-2):127–139, 1992.

[25] G. Lee and C. Scott. The one class support vector machine solution path. *ICASSP 2007. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2:II–521 – II–524, 2007.

[26] G. Loosli, G. Gasso, and S. Canu. Regularization paths for $\nu$-SVM and $\nu$-SVR. *ISNN, International Symposium on Neural Networks, LECTURE NOTES IN COMPUTER SCIENCE*, 4493:486, 2007.

[27] K. G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Number Chapter 5. 1988.

[28] K. Ritter. Ein verfahren zur lösung parameter-abhängiger, nicht-linearer maximum-probleme. *Unternehmensforschung*, 6:149–166, 1962.

[29] K. Ritter. On parametric linear and quadratic programming problems. *Mathematical Programming: Proceedings of the International Congress on Mathematical Programming. Rio de Janeiro, 6-8 April, 1981 / ed.: Richard W. Cottle, Milton Luiz Kelmanson, Bernhard H. Korte*, pages 307–335, 1984.

[30] S. Rosset and J. Zhu. Piecewise linear regularized solution paths. *Ann. Statist.*, 35(3):1012–1030, 2007.

[31] B. Schölkopf, J. Giesen, and S. Spalinger. Kernel methods for implicit surface modeling. 2004.

[32] A. Smola and B. Schölkopf. A tutorial on support vector regression. *NeuroCOLT2 Technical Report*, (NC2-TR-1998-030), 1998.

[33] J. Spjøtvold. Parametric programming in control theory. *PhD Thesis*, 2008.

[34] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[35] G. Wang. A new solution path algorithm in support vector regression. *IEEE Transactions on Neural Networks*, 2008.

[36] G. Wang, D. Yeung, and F. Lochovsky. Two-dimensional solution path for support vector regression. *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 993–1000, 2006.

[37] G. Wang, D.-Y. Yeung, and F. Lochovsky. A kernel path algorithm for support vector machines. *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007.